NASA/CR—2002-211979

Firewall Traversal for CORBA Applications Using an Implementation of Bidirectional IIOP in MICO

Robert I. Griffin
RS Information Systems, Inc., Brook Park, Ohio

November 2002

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2002-211979

Firewall Traversal for CORBA Applications Using an Implementation of Bidirectional IIOP in MICO

Robert I. Griffin
RS Information Systems, Inc., Brook Park, Ohio

Prepared under Contract NAS3–99175

National Aeronautics and
Space Administration

Glenn Research Center

November 2002

Acknowledgments

This research was conducted under NASA's CICT/CNIS Information Power Grid Task. The author would like to express his gratitude to Isaac Lopez and Greg Follen. Karel Gardas also helped in this work by suggesting the creation of the BiDirIIOP class. Finally, the author would like to thank Scott Townsend for the donation of various and sundry sage advice.

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

This report contains preliminary findings, subject to revision as analysis proceeds.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Firewall Traversal for CORBA Applications Using an Implementation of Bidirectional IIOP in MICO

Robert I. Griffin
RS Information Systems, Inc.
Brook Park, Ohio 44142
Robert.I.Griffin@grc.nasa.gov

The Object Management Group (OMG) (1) has added specifications to the General Inter-ORB Protocol (GIOP 1.2), specifically the Internet Inter-ORB Protocol (IIOP 1.2), that allow servers and clients on opposing sides of a firewall to reverse roles and still communicate freely (2). This addition to the GIOP specifications is referred to as Bidirectional GIOP.

The implementation of these specifications as applied to communication over TCP/IP connections is referred to as 'Bidirectional Internet Inter-ORB Protocol' or BiDirIIOP. This paper details the implementation and testing of the BiDirIIOP Specification in an open source ORB, MICO, that did not previously support Bidirectional GIOP (3). It also provides simple contextual information and a description of the OMG GIOP/IIOP messaging protocols.

Recommendations

- BiDirIIOP should be used in situations where servants located outside the firewall must request callbacks to clients behind the firewall.
- Additional effort should be expended to ensure full compliance with the OMG IIOP 2.6.1 specifications. For example, mechanisms for the denial of BiDirIIOP requests at the Portable Object Adapter level should be implemented.
- Research must be done to provide secure CORBA interactions between clients and servants that utilize BiDirIIOP. This will ensure that negative side effects, such as client masquerading, do not occur.

1. Background

The rationale for this research is driven by three major considerations. The first of these is NASA's drive to integrate engineering applications within the Information Power Grid (IPG) (4). The number of computational resources offered by IPG promises to increase the potential for integration of diverse engineering applications. This integration in turn creates the capability to perform and analyze complex, high-fidelity aerospace simulations in a quick and efficient manner.

The second consideration is the nature of the engineering applications that need to be integrated into the Information Power Grid. Many of the NASA aerospace engineering applications are sensitive and must be homed and executed in secure locales. Often there are non-disclosure agreements and distribution restrictions that accompany the use of such applications for research. Such agreements and restrictions preclude distribution and deployment of these applications on the Information Power Grid as the majority of IPG hosts are located outside of domain specific firewalls (e.g. NASA Glenn Research Center). If the drive is to integrate such applications with IPG and one is not able to execute them on IPG resources then it is perhaps better to offer them as resources available to users of NASA's Information Power Grid. This offering can be made in a manner such that non-authorized parties are unable to access or perform computations with the restricted applications.

The Common Object Request Broker Architecture (CORBA) is a middleware specification for creating distributed computing frameworks. NASA research shows that CORBA technology can be used to integrate sensitive applications within simulation environments such as the Numerical Propulsion System Simulation (NPSS) (5). This integration of applications and simulations can use direct linkages to CORBA routines within the source code (6)(7). Additionally, CORBA-based applications may offer a means to indirectly provide the application's functionality utilizing wrapping techniques that rely on scripts, file input/output, file signals or pipes (8)(9). The end

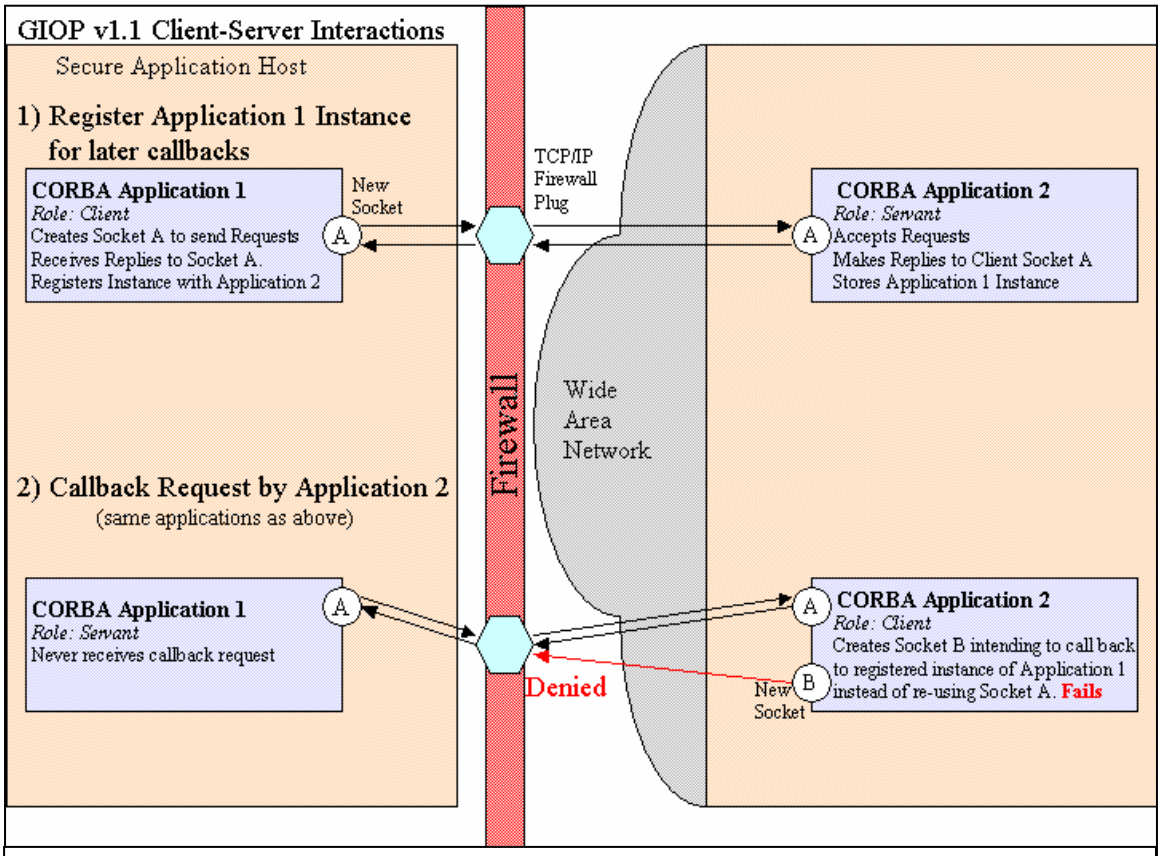

Figure 1: Interactions between clients and servants under GIOP version 1.1. *Internet Inter-ORB Protocol (IIOP) driven CORBA applications are not able to reverse client and servant roles and maintain communication across firewalls. The Object Management Group's GIOP v1.2 specification allows clients and servers to re-use TCP/IP connections when Bidirectional IIOP is specified.*

results are similar in that they make available engineering applications that would be otherwise unavailable.

The final consideration is that of network security. Specifically, the restrictions placed upon distributed computing environments when network administrators use firewalls to limit the access to secure domains. CORBA applications that are separated by firewalls may not function according to their original design or fail entirely.

MICO (3) is an open-source, C++-based implementation of the CORBA standards that is used for much of the research and development done with the Information Power Grid at Glenn Research Center (GRC). The MICO ORB (version 2.3.7) did not support the firewall traversal mechanisms, referred to as BiDirIIOP, that are recommended by the OMG CORBA v2.6.1 specification .

It is the goal of this work that BiDirIIOP be implemented in MICO. In realizing this goal application developers also moved a step closer to being able to provide sensitive engineering applications as Grid Resources while maintaining their position on a secure host located behind the firewall. Code created for the MICO implementation of BiDirIIOP could then be submitted to the open-source community as a contribution to their efforts.

1.1 Network Security and CORBA

The increasing need for Internet security has prompted NASA network administrators to limit the number of TCP/IP connections that are available for use between client and server applications that must talk across a Wide Area Network (WAN). The rule set that is used to either prohibit or allow communications between disparate domains is often referred to as a firewall. The increased use of firewalls has some profound effects on the interaction of applications that rely on CORBA.

Properly configured clients that reside behind the firewall may communicate with services outside of the firewall in a manner that renders the firewall transparent. Typically, their network administrators assign a single port to CORBA clients through which communications

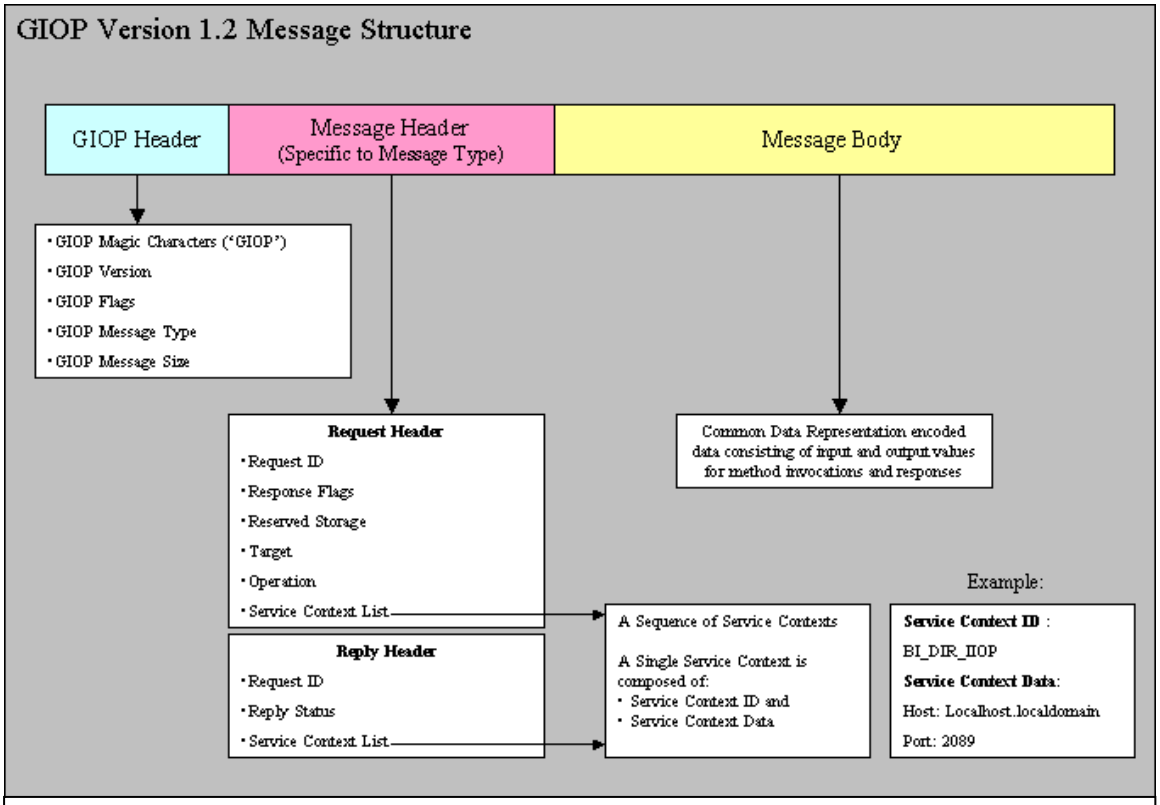

Figure 2: Components of GIOP Version 1.2 Messages. *GIOP messages consist of two Headers (GIOP and Message) as well as a Message Body. The Header Structures are very clearly defined by the OMG Specifications. All data transferred in a GIOP message is encoded according to the Common Data Representation (CDR) rules that further defines the encoding for all CORBA Interface Definition Language (IDL) Types. Also, shown is the structure for Service Contexts to be included in the Service Context Lists of GIOP Request and Reply Messages. The Bidirectional Service Context is provided as a sample.*

to the outside may occur. Servants located on the other side of the firewall must be configured to listen on that specific port. Failure to properly configure the base level client-server communications typically results in loss of functionality.

In the presence of firewalls, outside-of-the-firewall CORBA servants may respond appropriately to inside-of-the-firewall client requests, but may not initiate requests to those same clients. Problems occur when the typical client-server identities are reversed and communication must be initialized from an application outside of the firewall. This role reversal occurs when servants provide information asynchronously to clients and servants must use a callback to access one of the client's methods (Figure 1).

These problems may be traced to previous Object Management Group (OMG) specifications for the underlying messaging protocol: General Inter-ORB Protocol, or GIOP. GIOP minor version 1.0 and 1.1 clearly stated that connections were not symmetrical and that only clients can initialize connections and send requests and that only servers can accept and reply to connections. GIOP 1.2 relaxes these restrictions in instances where both clients and servers agree to share connections.

1.2 CORBA Communications

The OMG has established GIOP to define the method for encoding and decoding the data types of CORBA's interface definition language (IDL). GIOP is designed with simplicity, scalability and generality in mind. The Internet Inter-ORB Protocol (IIOP) may be viewed as the implementation of GIOP that uses TCP/IP as its transport and network layers.

GIOP messages define several types of messages: Request, Reply, LocateRequest, LocateReply, CancelRequest, CloseConnection, MessageError and Fragment. Each of these messages is composed of a GIOP header, a message header and a message body (Figure 2). The small number of message types maintains the simplicity of GIOP.

The OMG specification for Bidirectional GIOP v1.2 may be implemented for TCP/IP connections as Bidirectional IIOP (BiDirIIOP).

This has some profound changes on messages and the contents of messages that are sent between clients and servants.

One of the primary effects of the new specification is on which party (client or servant) can originate which types of messages. Both parties can send each of the eight types of GIOP messages when bidirectional GIOP specifications are followed. This contrasts with v1.0 and v1.1 specifications in that, under these specifications, clients were restricted to messages of type Request, CancelRequest, LocateRequest, MessageError and Fragment. Servants in these older specifications were limited to Reply, LocateReply, CloseConnection and Fragment message types. The change is profound in that it sets the stage for allowing the sort of role-reversal that might be required for CORBA applications that use callbacks.

Clients and servants also require some means by which they can announce their ability to make and receive bidirectional connections. The specification provides a means for this by creating the BiDirIIOP Service Context. This service context is identified by a service identification constant of BI_DIR_IIOP inserted into a list of other service contexts in the message header for GIOP request and reply message types (Figure 2). The inserted BiDirIIOP Service Context also includes a sequence of hostnames and ports. This sequence of hosts and ports allows message recipients to maintain a list of potential bidirectional connections and, when needed, to pair these open connections with message destinations.

The specification indicates that the Request ID found in the GIOP Message Headers should unambiguously associate requests and replies. This simply means that connection originators must have even numbered Request ID's. Likewise, responders must have odd numbered Request ID's.

Finally, the OMG specifies policy requirements for bidirectional communications. The inclusion of a bidirectional policy with a value of 'BOTH' is used to indicate that bidirectional communications can occur. Similarly, the default value of 'NORMAL' indicates that no such communications should occur. Furthermore, the Portable Object Adapter (POA) responsible for directing requests to CORBA servants should deny bidirectional communications from clients to servants whose stated BiDirPolicy is 'NORMAL'.

2. Tools

One ORB implementation was used on several different platforms for this study. It was also necessary to create a simple test scenario that could serve as an appropriate analog for the larger integrated infrastructure. Changes to the MICO ORB source code as well as source code for applications used in this study may be found at:

http://accl.grc.nasa.gov/rgriffin/bidiriiop/bidiriiop.html

2.1 ORB Implementation

The work reported here was conducted with MICO 2.3.7, an open-source, C++-based CORBA implementation. This is the most current public release of MICO and is distributed under the GNU General Public License (GPL). Open source was desired, as it is often necessary to make corrections to the source code or add features, as was the case with this work, to the code body.

MICO has a solid history of use in production grade integration (10) and has one of the most complete CORBA specification compliances (11). MICO also includes support for secure communications via SSL and is used in other NASA Glenn Information Power Grid projects.

Furthermore, MICO is available for many platforms. Among the operating systems and architectures supported are Linux, SGI, HPUX, and Windows. This diversity is necessary as the execution of many of the sensitive engineering applications that are used at NASA may be confined to powerful machines that meet the applications computational requirements, whereas the locale of a CORBA client can run the gamut of platforms.

2.2 Test Platforms

This research was conducted initially on a dual Pentium 3 x86 system running Linux RedHat 7.3 situated behind the NASA Glenn firewall. Later work extended to the Aeroshark Cluster, a 128-processor x86 cluster running Debian GNU/Linux (12) and Beowulf Clustering middleware. Finally, platforms used in this work also included Glenn Research Center's Sharp system: a 64-bit SGI Origin 2000 running IRIX 6.5.15.

MICO 2.3.7 builds for all platforms proceeded smoothly and although MICO could have been built with SSL-support, it was not.

```
#pragma prefix "nasa.gov"

interface Base {
  boolean doIt();
};

interface Master {
  boolean connect();
  boolean disconnect();
  boolean BaseRegister(in Base BaseServer);
  boolean BaseUnregister (in Base BaseServer);
  boolean Execute();
  boolean ExecuteBase(in string BaseName);
};
```

Listing 1: Simple Interface Definition. *This IDL formulation was used to simulate interactions of a more complex architecture in which engineering applications could be made available as Grid Resources.*

Furthermore, the MICO 2.3.7 build for Sharp was limited to the 32-bit Application Binary Interface (ABI).

2.3 Test Scenario

The test scenario had one key criterion. That was that a CORBA application located outside of the firewall had to be able to asynchronously communicate with another CORBA application located inside the firewall. This communication took the form of an unsolicited callback.

The scenario used to test the changes to the ORB Core was a simplified version of the overall architecture envisioned for providing engineering applications as Grid Resources. This simplification considered only three participants:

1) A *Base* servant located behind a firewall. This application was used to register, execute and provide the results of the potential engineering application to a *Master* servant. It was the recipient of method callbacks originated by the *Master* servant.
2) A *Master* servant located on the outside of a firewall served as the call site for engineering applications located within the firewall. It is the registration point for *Base* servants and the originator of the method callbacks to those instances.
3) A *Client* application that could contact the *Master* servant and request execution of engineering codes wrapped by the *Base* servant.

The interactions among these participants were formalized using CORBA's Interface Definition Language (IDL) (Listing 1). Method definitions were restricted to an execute method. for the Base servant. This method of the Base servant was the method designated to be the callback used by the Master servant to test asynchronous communications.

Connect/disconnect, register/unregister and execute/execute-by-name methods were assigned to the Master servant. The register method was used to provide the Master servant with a reference to a Base servant. The Master servant's execute method was used by the Client application to trigger requests for Base execution (i.e., the method callback).

More complicated interfaces could have been defined, but were deemed unnecessary in that the key elements for secure, Grid-enabled engineering applications could be extrapolated from the architecture defined by the IDL (Figure 3).

3. Changes to the MICO Source Code

The addition of bidirectional communications to the MICO Object Request Broker (ORB) requires that the following tasks be accomplished:

1) The ORB has to send the Bidirectional Service Context information in Request messages.
2) The ORB has to maintain a list of the information that it receives in the Bidirectional Service Context of incoming messages.
3) The ORB, when making connections, has to consult the list of Bidirectional information to determine whether or not to re-use a connection.

The changes to the MICO source code were confined to three units each of which was composed of header and implementation files. The ORB Core unit, (orb_mico.h and orb.cc) and the Portable Object Adapter (POA) unit

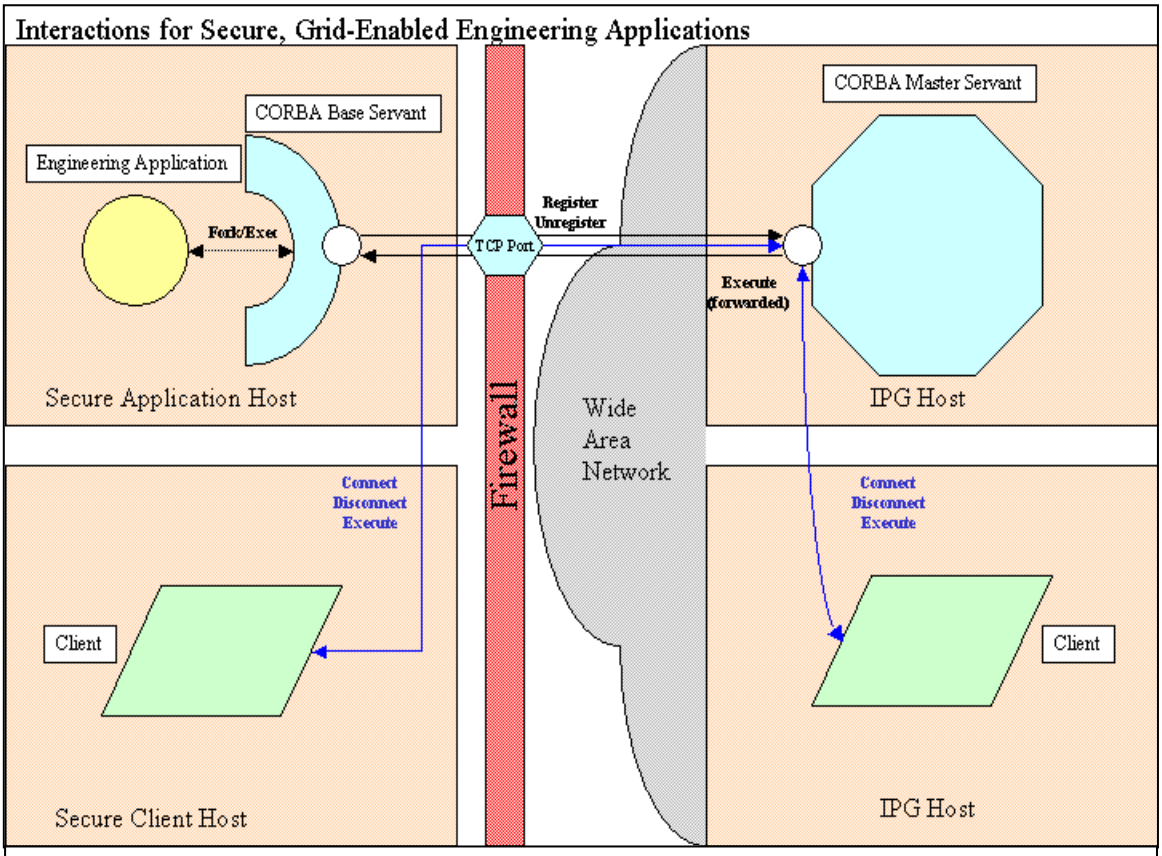

Figure 3: Simplified Architecture for providing Engineering Applications as Grid Services. *Interactions between Base and Master Servants are confined to Register/Unregister and Execute requests. Clients may connect to, disconnect from and request executions from the Master Servant. The Base Servant uses an indirect wrapping method (POSIX Fork/Exec) to begin the Engineering Application. The test architecture replaced the execution of an engineering application with 'execute' directions for the directory listing command 'ls > ls.out'.*

(poa_impl.h and poa_impl.cc) received few changes, whereas the Inter-ORB Protocol unit (iop.h and iop.cc) contained the greatest number of changes. Most of the minor changes to these units have been omitted in the interest of brevity.

3.1 Changes to the ORB Core Unit

One of the first priorities in this work was to create a simple command-line switch that would indicate that the ORB was to use bidirectional communications. Thus, the ORB was set to recognize a new command-line switch:

'**–ORBBiDirIIOP**'

Once recognized a private boolean member of the ORB class, **isBiDir**, is changed from its default value of *false* to *true*. Two public accessor functions for the **isBiDir** member were also created in the ORB class, **setBiDirectional** and **isBidirectional**.

The ORB Core unit was altered to include an Object Adapter: **bidiriiop_instance**, an instance of the new BiDirIIOP class (Section 3.3). This instance initially assigned a NULL-value is assigned a newly constructed BiDirIIOP class when **isBiDir** is set to *true*. Additionally, ORB GIOP and IIOP versions are assigned values of *1.2* to ensure compliance with the GIOP 1.2 standard. The **bidiriiop_instance** variable is then assigned IIOP listening points (i.e., host name and port number).

Changes were also made to the assignment of ORB message identification numbers. Bidirectional applications that take the role of the client by initiating communications with a CORBA servant are assigned even ORB message id's. Similarly, applications acting as the servant are assigned odd message id's. These changes are dictated by the OMG Bidirectional GIOP specifications and are necessary to allow interoperability between different ORB implementations. However, ORB-interoper-ability was not tested in the scope of this research.

3.2 Changes to the POA Unit

Changes to the Portable Object Adapter (POA) Unit provided a simple framework for the

```
CORBA::ORB_var orb;
orb = CORBA::ORB_init ( argc , argv );
CORBA::Object_var poaobj = orb->resolve_initial_references ( "RootPOA" );
PortableServer::POA_var poa = PortableServer::POA::_narrow ( poaobj );
PortableServer::POAManager_var mgr = poa->the_POAManager();
CORBA::Any polval;
PortableServer::POA_var bidirpoa;
polval <<= BiDirPolicy::BOTH;
CORBA::Policy_var pPol = orb->create_policy( BiDirPolicy::BIDIRECTIONAL_POLICY_TYPE,
                                             polval );
CORBA::PolicyList pl;
if (!CORBA::is_nil(pPol))
{
  pl.length(1);
  pl[0] = pPol;
  bidirpoa = poa->create_POA ("BiDirPOA", mgr, pl);
}
```

Listing 2: Code Listing for the creation of BiDirectional Portable Object Adapter. *Initially, a request to the ORB to create a policy of type BiDirPolicy::BIDIRECTIONAL_POLICY_TYPE with a value of BiDirPolicy::BOTH is made. This policy is then Passed to the RootPOA as part of a request request for a new Portable Object Adapter. It is this new POA that is responsible for intercepting and allowing or denying bidirectional communications.*

detection of bidirectional communication policies. Internally, two new private members were added to the **POA_impl** class. The first of these additions was the boolean variable **isBiDir**. The second of these was **bidir_policy**; an instance of **BiDirPolicy::BidirectionalPolicy**.

Changes were made to the **set_policies** member function of the **POA_impl** class to create a new policy, **bidir_policy**, whose value was initially NULL and later changed according to the existence and/or value assigned to the policies that were passed in as arguments to **set_policies**. If the arguments do not contain a **BiDirPolicy::BidirectionalPolicy** then the default value of BiDirPolicy::NORMAL for **bidir_policy** is used.

The Root POA always has a BiDirPolicy value of NORMAL. POA's that are created by servant applications may be assigned a BiDirPolicy of BOTH. A BiDirPolicy is requested from the ORB's **create_policy** method. This policy is then propagated to new POA's *via* the RootPOA's **create_POA** method (Listing 2).

It is anticipated that an interceptor for the POA to allow or deny incoming bidirectional communications will be added to the source code. This, however, was not implemented in the current research. Instead, bidirectional communications are monitored only in the IOP unit.

3.3 Changes to the IOP Unit.

The changes to the Inter-ORB Protocol (IOP) Unit were the most extensive. Two classes, **GIOPConn** and **GIOPCodec**, were modified and a new class, **BiDirIIOP**, was created. **BiDirIIOP** controls many instances of **GIOPConn** each of which has an instance of **GIOPCodec**.

Two global variables were also added to the IOP unit. These variables were of type **ListenPointList**. **ListenPointList** is a list containing multiple, different instances of the structure **ListenPoint**. Each **ListenPoint** is composed of a **Host** string and a **Port** integer. The **_lpServer** instance of the listen point list is responsible for reporting the ORB-assigned listen points in the Bidirectional Service Context data of outgoing GIOP messages (Figure 2). A list of potential bidirectional communications is maintained in **_lpClient**. The **_lpClient** instance is filled using information from the Service Context data of incoming GIOP messages (i.e., the values contained in the client's **_lpServer**).

3.3.1 Changes to GIOPCodec and GIOPConn

GIOPCodec is responsible for encoding and decoding GIOP messages. It is the point where the BiDirIIOP Service Context can be detected in the GIOP Message Header of incoming messages and placed in GIOP Message Header of outgoing messages.

GIOPCodec was modified to include a private boolean member **isBiDir**. This boolean member is only set during the initial construction of the class. Its default value is *false* unless otherwise specified. It is used in a modified member function, **get_contextlist**, to determine whether an attempt should be made to detect the Bidirectional Service Context in incoming messages. The global variable **_lpClient i**s modified to include the ancillary data when a Bidirectional Service Context is identified. The variable, **isBiDir,** is also used to determine whether a Bidirectional Service Context should be inserted into outgoing messages.

GIOPCodec member function **put_contextlist** was modified to perform this insertion. The data inserted into the context mirrors the contents of **_lpServer**.

Changes to **GIOPConn** were much simpler. TCP/IP transport is flagged as blocked according to the static member function **isblocking()** of **BiDirIIOP** as opposed to that of the older class **IIOPProxy** (Section 3.3.2). Choice of blocking rules is determined when a bidirectional flag in the constructor is set to true. This is necessary as the **isblocking()** functions of **BiDirIIOP** and **IIOPProxy** are indicative of the underlying state and behavior of two very different classes.

3.3.2 Creation of the BiDirIIOP Class

The components for a bidirectional object adapter existed in the MICO code prior to this work. All that was missing was their integration into a cohesive class. Two classes, **IIOPProxy** and **IIOPServer** are subclassed from **CORBA::ObjectAdapter**. Each of these classes implements the virtual abstract methods that are defined in the base **CORBA::ObjectAdapter** class.

IIOPProxy is used as the channel by which asynchronous request, bind and locate messages are propagated to remote ORBS. Conversely, **IIOPServer** is used to direct incoming messages to the appropriate objects contained within the local ORB and to propagate the responses of those objects back to remote clients.

One key limitation to providing bidirectional communications with the pre-existing architecture was that each of these derived classes maintained its own private set of **GIOPConn** instances. Thus, it was not possible to map the connection addresses of incoming messages (**IIOPServer**) with those of the outgoing (**IIOPProxy**) messages. Connection reuse was implausible in the light of this fundamental disconnect.

The **BiDirIIOP** class created a solution to this problem by combining **IIOPProxy** and **IIOPServer**. Server and proxy connections are kept in separate lists. Only slight overlap occurs in terms of the member functions (e.g., events that closed or cancelled connections in **GIOPConn**-based callback). A new member function, **isServerConn**, is used to differentiate between connections belonging to the proxy or server pools.

Connection information is shared and mapped to hosts and ports efficiently in this new class. The new class defines and maintains a new member variable **_bidir_conns**. The **_bidir_conns** variable represents a mapping of connection string to **GIOPConn** (type **MapStringConn**). The connections string is composed of the protocol ('inet') the hostname and the port (e.g., 'inet:localhost:2089'). This information (host and port) can be found in **_lpClient** after an incipient call to **GIOPCodec::get_contextlist** occurs.

Maintenance for **_bidir_conns** requires both addition and deletion of entries as GIOP connections are added and subtracted from **BiDirIIOP.** Entries are added after GIOP request and reply messages are processed and are removed from the **_bidir_conns** map when a GIOP CloseConnection is received. Finally, the contents of **_bidir_conns** are deleted entirely during cleanup of the **BiDirIIOP** class.

The list of connections in **_bidir_conns** can thus be consulted to find connections (i.e. **GIOPConn**'s) for reuse. This consultation occurs in the **BiDirIIOP**'s **make_conn** member function that is used by the asynchronous **invoke, bind** and **locate** calls on a remote object.

Remote objects are contacted *via* an Interoperable Object Reference (IOR). IOR profiles for these remote objects contain a target address representing the address that the remote object's ORB is listening on. This 'listening' address is contained in the remote ORB's **_lpServer** list and is 'advertised' by the object in the Bidirectional Service Context of every **BiDirIIOP** request.

Not by coincidence, this IOR Profile address, when stringified, is the same format as the connection string contained in **_bidir_conns**. The **GIOPConn** associated with a **_bidir_conns** entry may thus be reused for bidirectional communication when the stringified IOR profile's address matches the connection string. Searches through **_bidir_conns** are conducted iteratively, but further work can be done to implement a fast find function for this map.

If a reusable connection cannot be found then the **BiDirIIOP::make_conn** function first looks for the connection in its list of proxy connections. If a proxy connection cannot be found in this list then the function reverts to creating a new connection to the remote object. This behavior is indistinguishable from that of the **IIOPProxy::make_conn** function.

Debugging messages were also added to the BiDirIIOP class so that its reuse of connections could be verified. These debugging messages could be enabled using the command-line flags: '**-ORBDebug IIOP –ORBDebug GIOP**'.

```
1  MasterImpl -ORBDebug IIOP -ORBDebug GIOP -ORBIIOPAddr inet:sharp.ipgdomain:65435 \
   -ORBBiDirIIOP
   IIOP: BIDIR server listening on inet:sharp.ipgdomain:65435 IIOP version 1.2

2  IIOP: BIDIR new connection opened from inet:firewall3.nasadomain:65433
   IIOP: BIDIR incoming data from inet:firewall3.nasadomain:65433
   GIOP: incoming CodeSets context
   GIOP: requested TCS-C is ISO 8859-1:1987; Latin Alphabet No. 1
   GIOP: requested TCS-W is ISO/IEC 10646-1:1993; UTF-16, UCS Transformation Format
   16-bit form
   GIOP: BiDirectional IIOP context_id received
   GIOP: BIDIR incoming Request from inet:firewall3.nasadomain:65433 with msgid 2
   GIOP: BIDIR sending Reply to inet:firewall3.nasadomain:65433 for msgid 2 status is 0
   IIOP: BIDIR incoming data from inet:firewall3.nasadomain:65433
   GIOP: BiDirectional IIOP context_id received
   GIOP: BIDIR incoming Request from inet:firewall3.nasadomain:65433 with msgid 4
   GIOP: BIDIR sending Reply to inet:firewall3.nasadomain:65433 for msgid 4 status is 0

3  IIOP: BIDIR new connection opened from inet:aeroshark.ipgdomain:33624
   IIOP: BIDIR incoming data from inet:aeroshark.ipgdomain:33624
   GIOP: incoming CodeSets context
   GIOP:   requested TCS-C is ISO 8859-1:1987; Latin Alphabet No. 1
   GIOP:   requested TCS-W is ISO/IEC 10646-1:1993; UTF-16, UCS Transformation Format
   16-bit form
   GIOP: BiDirectional IIOP context_id received
   GIOP: BIDIR incoming Request from inet:aeroshark.ipgdomain:33624 with msgid 2

4  IIOP: BIDIR reusing GIOP connection to inet:firewall3.nasadomain:65433
   GIOP: BIDIR sending Request to inet:firewall3.nasadomain:65433 msgid is 4
   IIOP: BIDIR incoming data from inet:firewall3.nasadomain:65433
   GIOP: BiDirectional IIOP context_id received

5  GIOP: BIDIR sending Reply to inet:aeroshark.ipgdomain:33624 for msgid 2 status is 0
   IIOP: BIDIR connection to inet:aeroshark.ipgdomain:33624 closed or broken

6  IIOP: BIDIR incoming data from inet:firewall3.nasadomain:65433
   GIOP: BiDirectional IIOP context_id received
   GIOP: BIDIR incoming Request from inet:firewall3.nasadomain:65433 with msgid 6
   GIOP: BIDIR sending Reply to inet:firewall3.nasadomain:65433 for msgid 6 status is 0
   IIOP: BIDIR connection to inet:firewall3.nasadomain:65433 closed or broken
```

Listing 3: Debug output from the Master Servant. **(1)** The Master servant is started and listens on its assigned port. **(2)** Communications with the Base servant shows **connect** and **register** requests. **(3)** Similarly, the client application performs a **connect** request with the Master. Codeset negotiations are incidental for all GIOP version 1.2 connections. The final request from the client is for the execution of the Base servant. **(4)** The Master servant immediately reuses the Base Servant connection (bold) with a callback request for the Base Servant's **doit** method. **(5)** The reply from the Base Servant is forwarded to the client application and connections are closed by the client application (close connection). **(6)** Subsequently, the Base servant makes calls to the Master's **unregister** method and closes the connection. Domain names and the name of the secure hosts have been changed to protect NASA resources.

4. Results

Bidirectional communications testing of the CORBA applications utilizing the new **BiDirIIOP** class was performed on three different hosts. Two of these hosts were the IPG Resources: Aeroshark Cluster and Sharp. The last host was located on a Linux machine running inside the NASA Glenn firewall. Each of the elements of the simple testing architecture (Figure 3) was placed on a different host. The choice of hosts also ensured that the changes made to the code to allow bidirectional communications would be compilable on multiple platforms and could thus be integrated quickly into the MICO concurrent versioning system. The breakdown is as follows:

1) Secure Host: Home for the Base Servant (*BaseImpl*).
2) Sharp: Home for the Master Servant (*MasterImpl*).
3) Aeroshark: Home for the Client Application (*miniclient*).

The command-line parameters arguments are of the form:

```
<appname> \
        -ORBDebug IIOP \
        -ORBDebug GIOP \
        -ORBIIOPAddr \
                inet:<host>:<port> \
        -ORBBiDirIIOP
```

Execution for this test took place in four steps. First, the Master servant was started. Subsequently a file containing the Master servant's Interoperable Object Reference (IOR) was copied to both the Base servant and client machines. The Base servant was started next. Finally, the client application was executed.

The Master servant's debugging output for this suite of executions clearly indicated that bidirectional communications and connection reuse has occurred. Furthermore, these results have shown that communications have gone through the firewall (firewall3) in both directions (Listing 3).

5. Conclusions

The initial implementation of the OMG Bidirectional Inter-ORB Operability protocol works well in mixed (secure and insecure) domain testing. It is anticipated that the MICO BiDirIIOP implementation will be useful in crossing (and re-crossing) NASA's secure network boundaries. This work will be incorporated in the effort to provide engineering applications as resources to the NASA Information Power Grid.

There are several considerations for future studies with BiDirIIOP. The first of these is the means by which POA's are to permit or deny bidirectional connections. The inclusion of the bidirectional policy is only the first step in this process. The OMG specification leaves unclear the means by which individual POA instances are to check the bidirectional nature of incoming requests. One solution to this problem is to create an interceptor for each newly instantiated POA. The OMG has recently added separate specification for portable interceptors (1). However, MICO does not currently support this specification. Addition of portable interceptors to MICO could be helpful in implementing the functionality for BiDirIIOP implied by the specification. Changes to the IOP unit's **GIOPCodec** class would also be needed as the **GIOPCodec::get_contextlist** function currently removes the Bidirectional Service Context from the message header.

One further consideration is that of the security of bidirectional communications. The implementation of BiDirIIOP discussed within this paper does not prevent potentially foreign clients from masquerading as the recipient of a callback. A solution to this problem is to use Secure Socket Layer IOP (SSLIOP) as the OMG specification states that BiDirIIOP should not interfere with SSLIOP. However, no testing has been done to confirm that MICO implementation of BiDirIIOP works with SSLIOP.

A final consideration for future study is how the MICO implementation of BiDirIIOP affects other services such as the Event, Naming and Trader services. One desired scenario is that these services will actually perform across the NASA secure domains when the execution of these standalone services includes the bidirectional communication directive. This would be especially useful to the Event service. A bidirectional Event Service would allow callbacks to be made across the firewall, thus permitting communicating the occurrence of important events to a client situated on a secure host.

References

1. URL: http://www.omg.org.
2. Object Management Group. *The Common Object Request Broker: Architecture and Specification.* Revision 2.6.1. May, 2002.
3. A. Puder and K. Romer. MICO is CORBA. URL: http://www.mico.org.
4. URL: http://www.ipg.nasa.gov.
5. Lopez, G. J. Follen, R. Gutierrez, I. Foster, B. Ginsburg, O. Larsson, S. Martin, S. Tuecke, D. Woodford. NPSS on NASA's IPG: Using CORBA and Globus to Coordinate Multidisciplinary Aeroscience Applications. *Proceedings of the NASA HPCC/CAS Workshop,* Feb. 15-17, 2000. URL: http://accl.grc.nasa.gov/IPG/CORBA/NPSS_CAS_paper.html.
6. Sang, C. M. Kim, I. Lopez. *Developing CORBA-Based Distributed Scientific Applications from Legacy Fortran Programs.* URL: http://www.ipg.nasa.gov/research/papers/CAS_corba.pdf
7. S. Townsend. *Wrapping ADPAC CFD Code*. URL: http://accl.grc.nasa.gov/IPG/CORBA/wrap_fortran.scott.html.
8. Sang. C. Kim. *Developing a wrapping tool for CORBA application of FORTRAN codes*. URL: http://accl.grc.nasa.gov/IPG/CORBA/wrap_fortran.chan.html.
9. R. Gutierrez. *Techniques for Wrapping Fortran Codes.* URL: http://accl.grc.nasa.gov/IPG/CORBA/wrap_fortran.richard.html.
10. Weather Channel. URL: http://linuxtoday.com/news_story.php3?ltsn=2000-10-10-011-04-PR.
11. CORBA ORB Compliance. URL: http://www.opengroup.org/press/7jun99_a.htm.
12. Aeroshark. URL: http://cict.grc.nasa.gov/ashark/.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 2002	Final Contractor Report

4. TITLE AND SUBTITLE

Firewall Traversal for CORBA Applications Using an Implementation of Bidirectional IIOP in MICO

5. FUNDING NUMBERS

WU–704–40–24–00
NAS3–99175

6. AUTHOR(S)

Robert I. Griffin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

RS Information Systems, Inc.
2001 Aerospace Parkway
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER

E–13648

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2002-211979

11. SUPPLEMENTARY NOTES

Project Manager, Isaac Lopez, Vehicle Technology Directorate, NASA Glenn Research Center, organization code 0300, 216–433–5893.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 61 Distribution: Nonstandard

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The Object Management Group (OMG) has added specifications to the General Inter-ORB Protocol (GIOP 1.2), specifically the Internet Inter-ORB Protocol (IIOP 1.2), that allow servers and clients on opposing sides of a firewall to reverse roles and still communicate freely. This addition to the GIOP specifications is referred to as Bidirectional GIOP. The implementation of these specifications as applied to communication over TCP/IP connections is referred to as 'Bidirectional Internet Inter-ORB Protocol' or BiDirIIOP. This paper details the implementation and testing of the BiDirIIOP Specification in an open source ORB, MICO, that did not previously support Bidirectional GIOP. It also provides simple contextual information and a description of the OMG GIOP/IIOP messaging protocols.

14. SUBJECT TERMS

CORBA; MICO; Information power grid; Grids

15. NUMBER OF PAGES

16

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	